**document.getElementById**

**document.getElementsByTagName**

区别

1、ID前面必须是document，而获取TagName前面即可以跟document，也可以跟别的元素

2、ID只能找一个，而TagName找到的是一堆元素，所以访问的时候必须和个中括号

3、ID是静态方法，在哪一行没有找到就找不到了，而tagName是动态方法，如果下面加了东西，后面访问实际上是能找到的

**循环要帮我们解决什么样的问题？**

1、重复执行某些代码

2、每次执行的时候，有个数字在变化

如果符合这两点，我们就可以用上循环

如果弹出两千个1，键盘按坏了，还按不完，这时，我们就要让程序帮我们来做这个事了

var i = 0;

for ( ; i<3; ) {

alert(1);

i++;

}

**过程**

1) var i=0;

2) i<3; 关键！！！！

3) 括号里面的所有代码

4) i++

写大家都熟悉的方式

for(var i=0; i<3; i++){

// alert(i);

}

// alert(i); // 3

**for循环的应用**

找document.getElementsByTagName的每个元素

aLi[0].onclick = function (){ alert(1); };

aLi[1].onclick = function (){ alert(1); };

aLi[2].onclick = function (){ alert(1); };

给li添加今天明天后天，这要用上一个数组

即可以加点击事件，又可以加文字

**实例1：生成200个按钮**

document.body.innerHTML += '<input type="button" value="按钮" />';

生成200个，生成两千个，这就影响性能

说明：本身for不没有什么性能问题，问题就在innerHTML这里，说说为什么慢的过程，每次都要去document.body找东西，加东西，生成，再找东西，加东西生成，

**实例2，计算元素的座标，**

更改left，top，动态添加座标

练习：

1、生成100个div

2、来一个小v字形

**for套for，**

程序语句很多都是可以相互嵌套的，如if。

数组里可以放什么？可以放文字，可以放数字，还可以放数组。

var arr = [

[ 1,2,3 ],

[ 4,5,6 ],

[ 7,8,9 ]

];

for( var i=0; i<arr.length; i++ ){

// arr[i][0]

// arr[i][1]

// arr[i][2]

for( var j=0; j<arr[i].length; j++ ){

// alert( arr[i][j] );

}

}

为什么用j，行为规范

// for( var i=0; ){}

// for( var i=0; ){}

这样就可以都用i

**实例，模仿QQ**

<ul id="list">

<li>

<h2>我的好友</h2>

<ul>

<li>莫涛</li>

<li>张森</li>

<li>杜鹏</li>

</ul>

</li>

<li><ol><li>no</li><li>no</li><li>no</li></ol></li>

<li>

<h2>我的坏友</h2>

<ul>

<li>莫小涛</li>

<li>张小森</li>

</ul>

</li>

<li><ol><li>no</li><li>no</li><li>no</li></ol></li>

<li>

<h2>我的黑名单</h2>

<ul>

<li>莫张</li>

</ul>

</li>

</ul>

window.onload = function (){

var oUl = document.getElementById('list');

var aUl = oUl.getElementsByTagName('ul');

var len = aUl.length;

var aLi = null; // 空

for( var i=0; i<len; i++ ){

aLi = aUl[i].getElementsByTagName('li');

for( var j=0; j<aLi.length; j++ ){

aLi[j].style.border = '1px solid red';

}

}

};

**总结：我们的这些操作，归根结底，是给元素添加行间样式，**引出

oDiv.style.cssText = ' width:200px; height:200px; ';它实际上是在元素行间加，而不是改的css，要说明这点，注意cssText的写法，没有任何的兼容性问题

思考：改的是行间的，还是css样式表里的

有点像innerHTML替换的作用。

如：

oDiv.onclick = function (){

// oDiv.style.width = '200px';

oDiv.style.cssText = ' width:200px; height:200px; ';

};

oBtn.onclick = function (){

// oDiv.style.width = '100px';

oDiv.style.cssText = '';

};

**例子，自动生成一组新闻，**

根据新闻生成五个li，利用一个开关

<input id="btn1" type="button" value="自动生成5条新闻" />

<ul id="list" style="border:1px solid red;"></ul>

var oBtn = document.getElementById('btn1');

var oUl = document.getElementById('list');

var arr = [

'山西省委附近多次爆炸 官方称尚不确定是恐怖袭击',

'甘肃张掖明令禁止转基因 书记:无力辨别只能禁止',

'多地制定雾霾预案限行限排被批治标不治本',

'韩媒抱怨中国雾霾侵袭韩国 称其为"黑色灾难" ',

'伊朗革命卫队高官在叙利亚当"志愿者"被杀(图)'

];

oBtn.onclick = function (){

for( var i=0; i<arr.length; i++ ){

oUl.innerHTML += '<li>' + arr[i] + '</li>';

}

}

**解决添加多个的问题**

1、oBtn.disabled = true;禁用按钮

2、oBtn.style.display = 'none';隐藏按钮

都是按钮上面的，这都不是最好的，用户体验不好，想和户是怎么感觉

3、先清空，再生成，

特点：简单，浏览器受不了，性能差点

4、用判断，需要思考整个程序的执行顺序，把开关放不同的位置，看看执行的情况

**this**

**一、**

alert( this );

window 是 JS “老大”

window.alert( this );

谁让我们这个alert动了，它里面的this就指向谁

**二、函数里的this**

1、

function fn1(){

alert( this ); // window

}

// fn1();

// window.fn1();

理解：this就是调用当前方法或函数的哪个对象。谁给我糖，我跟谁走

2、

按钮点击的方式：

var oBtn = document.getElementById('btn1');

// oBtn.onclick = fn1;

oBtn.onclick = function (){

// this

fn1();它前面没有东西，但是其实前面是有个老大window的

};

3、

<input id="btn2" type="button" onclick="alert(this) fn1(); " value="按钮2" />

这里面的this

**总结：分四种情况**

function fn1(){

// this

}

1、 fn1(); this => window

2、 oDiv.onclick = fn1; this => oDiv

3、

/\*

oDiv.onclick = function (){

this ?

fn1(); fn1() 里的this => window

};

4、这种情况很少用，任何公司都不再使用这种写法

<div onclick=" this fn1(); "></div> fn1(); 里的 this 指的是 window

this的例子2个

例一，更改按钮的背景色

1、直接调用：this.style.background = 'yellow';

2、间接调用，调用一个命名函数，要用This来存一下

3、aBtn[i].onclick = fn1;

var aBtn = document.getElementsByTagName('input');

var that = null; // 空

for(var i=0; i<aBtn.length; i++){

/\*

aBtn[i].onclick = function (){

// this.style.background = 'yellow';

that = this;

fn1();

};

\*/

aBtn[i].onclick = fn1;

}

function fn1(){

// this => window

// that.style.background = 'yellow';

// this.style.background = 'red';

}

<input type="button" value="按钮1" />

<input type="button" value="按钮2" />

<input type="button" value="按钮3" />

做个东西：

练习